**Load Sample Text Data**

**Step-by-Step Instructions**

1. **Open Google Colab**
   * Go to Google Colab and create a new notebook by selecting **File > New Notebook**.
2. **Load Sample Data**
   * For this exercise, let’s use a short text paragraph as sample data. Alternatively, participants can load a text file or download a sample dataset (e.g., from Kaggle or the web).

python

# Sample text data

*text\_data = """*

NLP is fascinating! It's widely used in applications like chatbots, sentiment analysis, and machine translation.

People express emotions online with text, emojis 😊, and abbreviations like LOL or ASAP. But, the data needs preprocessing!

*"""*

1. **Display Raw Data**
   * Print the raw text data to observe its initial structure and characteristics.

python

*print("Raw Text Data:\n", text\_data)*

**Expected Output:**

vbnet

Raw Text Data:

NLP is fascinating! It's widely used in applications like chatbots, sentiment analysis, and machine translation.

People express emotions online with text, emojis 😊, and abbreviations like LOL or ASAP. But, the data needs preprocessing!

**Implement Basic Text Preprocessing Steps**

**Tokenization**

**Step-by-Step Instructions**

1. **Import NLTK and Perform Tokenization**
   * Use word\_tokenize for word-level tokenization and sent\_tokenize for sentence-level tokenization.

python

*import nltk*

*nltk.download('punkt')*

*from nltk.tokenize import word\_tokenize, sent\_tokenize*

# Sentence Tokenization

*sentences = sent\_tokenize(text\_data)*

*print("Sentence Tokenization:\n", sentences)*

# Word Tokenization

*words = word\_tokenize(text\_data)*

*print("\nWord Tokenization:\n", words)*

1. **Explanation**
   * **Sentence Tokenization**: Splits text into sentences, useful for analyzing sentence-level features.
   * **Word Tokenization**: Breaks down text into words, which helps to analyze individual words or tokens.

**Expected Output:**

less

Sentence Tokenization:

['NLP is fascinating!', "It's widely used in applications like chatbots, sentiment analysis, and machine translation.", ...]

Word Tokenization:

['NLP', 'is', 'fascinating', '!', 'It', "'s", 'widely', 'used', 'in', 'applications', ...]

**Removing Stop-Words and Punctuation**

**Step-by-Step Instructions**

1. **Import Stop-Words and Remove Them**
   * Use NLTK’s list of stop-words to filter out commonly used words.

python

*from nltk.corpus import stopwords*

*import string*

*nltk.download('stopwords')*

*stop\_words = set(stopwords.words('english'))*

# Removing Stop-Words and Punctuation

*words\_no\_stop = [word for word in words if word.lower() not in stop\_words]*

*words\_no\_punct = [word for word in words\_no\_stop if word not in string.punctuation]*

*print("\nText after Removing Stop-Words and Punctuation:\n", words\_no\_punct)*

1. **Explanation**
   * **Stop-Words Removal**: Filters out common words like “the,” “is,” “and” which don’t add meaning to the text.
   * **Punctuation Removal**: Removes punctuation, focusing on meaningful words only.

**Expected Output:**

less

Text after Removing Stop-Words and Punctuation:

['NLP', 'fascinating', 'widely', 'used', 'applications', 'like', 'chatbots', ...]

**Stemming and Lemmatization**

**Step-by-Step Instructions**

1. **Apply Stemming**
   * Use NLTK’s Porter Stemmer to reduce words to their root forms.

python

from nltk.stem import PorterStemmer

ps = PorterStemmer()

stemmed\_words = [ps.stem(word) for word in words\_no\_punct]

print("\nStemmed Words:\n", stemmed\_words)

1. **Apply Lemmatization with spaCy**
   * spaCy’s lemmatizer is more accurate as it considers context.
   * First, install spaCy (!pip install spacy) and download its English model (!python -m spacy download en\_core\_web\_sm).

python

*import spacy*

*nlp = spacy.load("en\_core\_web\_sm")*

*lemmatized\_words = [token.lemma\_ for token in nlp(' '.join(words\_no\_punct))]*

*print("\nLemmatized Words:\n", lemmatized\_words)*

1. **Explanation**
   * **Stemming**: Simplifies words to their root forms but may lead to slightly inaccurate forms.
   * **Lemmatization**: Generates the base form of words, accurately considering context.

**Expected Output:**

less

Stemmed Words:

['NLP', 'fascin', 'wide', 'use', 'applic', ...]

Lemmatized Words:

['NLP', 'fascinating', 'widely', 'use', 'application', ...]

### **Compare Raw and Processed Text**

**Objective:** Highlight the impact of preprocessing steps on the raw text data.

**Step-by-Step Instructions**

1. **Display Original vs. Preprocessed Text**
   * Show the initial raw text and final preprocessed words after removing noise, stop-words, and applying lemmatization.

python

*print("Original Text:\n", text\_data)*

*print("\nFully Processed Text:\n", ' '.join(lemmatized\_words))*

1. **Discussion**
   * Explain how preprocessing reduces irrelevant information, standardizes text, and makes it more accessible for NLP models to analyze.
   * Engage participants by asking them to observe specific changes, such as removed words or transformations (e.g., “running” to “run”).

**Expected Output:**

vbnet

Original Text:

NLP is fascinating! It's widely used in applications like chatbots, sentiment analysis, ...

Fully Processed Text:

NLP fascinating widely use application chatbot sentiment analysis machine translation people ...